Colas de Prioridad y Heaps

Pablo Castro
Algoritmos I-UNRC

Colas de Prioridad

Principales características:

- Se comportan como una cola, pero
- Los elementos tienen una prioridad,
- Los primeros en salir son los que más prioridad tienen.

Utiles para diferentes tareas, planificación de tareas, optimización del uso de la red, etc.

TAD Cola de Prioridad

- Elementos: una colección de elemento con un orden, o prioridades,

- Operaciones:
    - insertar (encolar): inserta un elemento en la cola,
    - DelMin (desencolar): elimina el elemento con la prioridad más chica (o más grande) de la cola.

La idea es hacer estas operaciones lo más eficientes posibles

Posibles Implementaciones

Podemos implementar las colas de prioridad de diferente formas:

- **Con arreglos ordenados:** DelMin es O(n) y el insert esO(log n + n)
- **Con listas enlazadas:** DelMin es O(n) y el insert es O(1).
- **Con listas enlazadas ordenadas:** DelMin es O(1) y insert es O(n)

Utilizando árboles se pueden lograr implementaciones más eficientes!

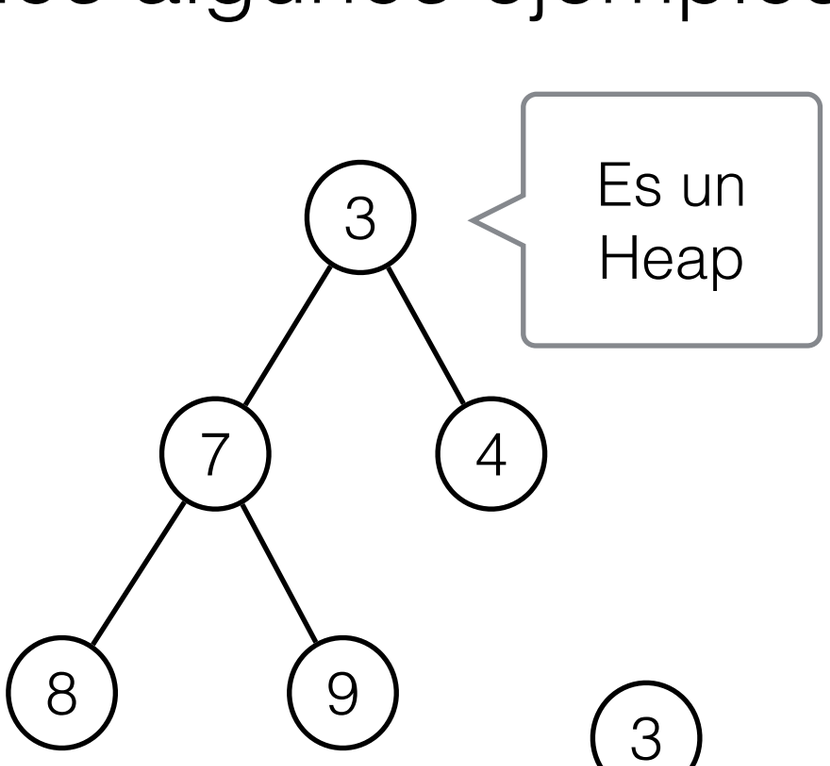
Heaps

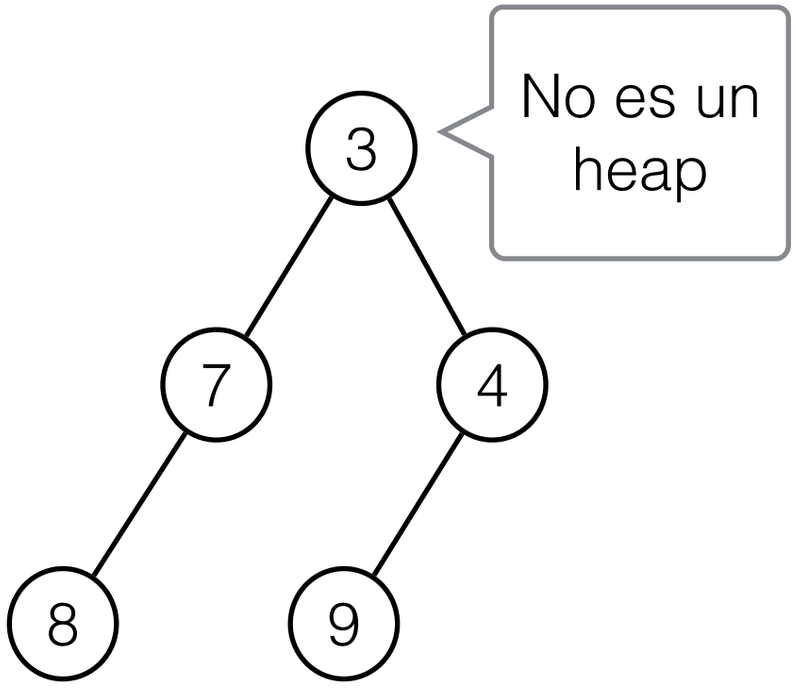
Los heaps son una implementación eficiente y elegantes de colas de prioridad:

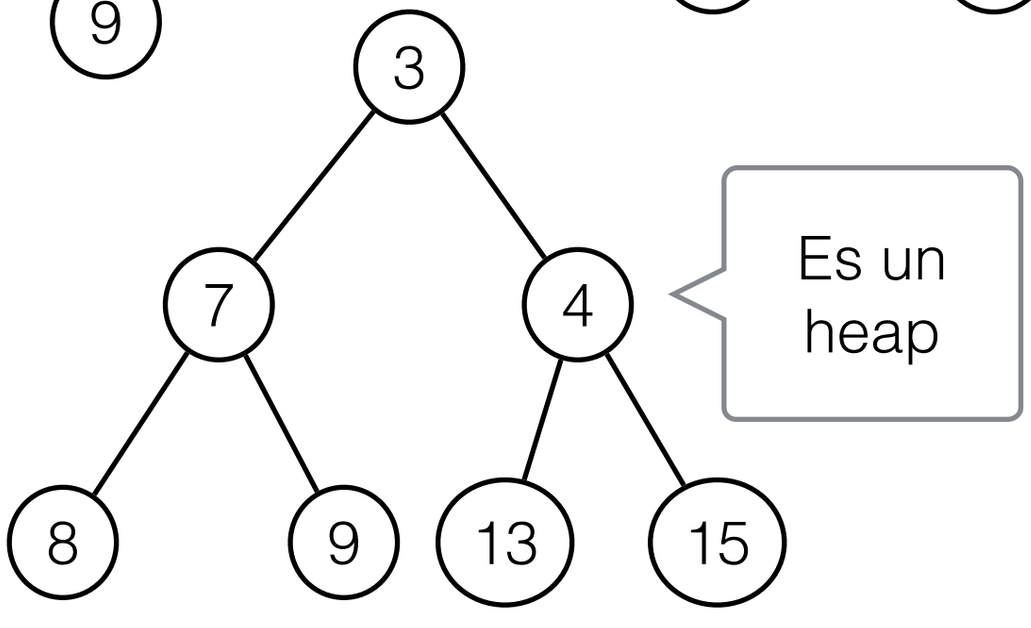
- Son árboles binarios,
- La raíz es más chica que los hijos, y esta propiedad se cumple recursivamente,
- Son completos, cada nivel tiene todos los nodos, excepto el último en donde pueden faltar algunos nodos a la derecha,
- La altura de un Heap es O(log n)

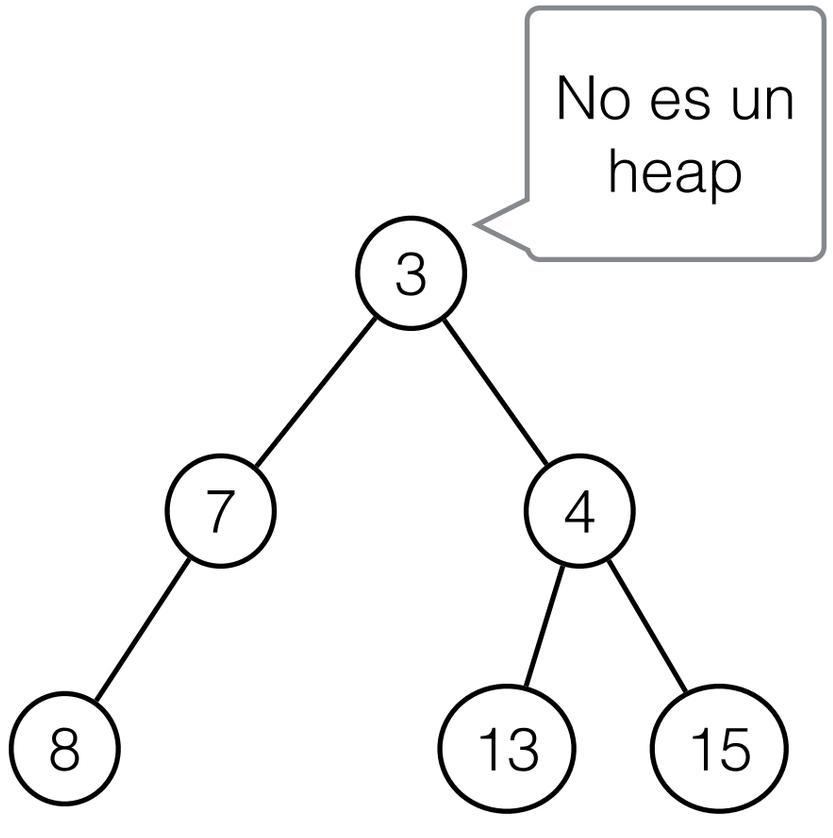
Algunos Ejemplos

Veamos algunos ejemplos:









Borrar en un Heap

- El mínimo está en la raíz, lo borramos,
- Quedan dos subárboles que son heaps, hay que reemplazar la raíz,
- Tomamos la hoja más a la derecha y la ponemos como raíz, todavía no tenemos un heap!
- Si hay algún hijo de la raíz más chico lo intercambiamos, y repetimos el procedimiento.

La altura es O(log n), entonces tenemos a lo sumo log n intercambios

Insertar en un Heap

La estrategia para insertar es parecida:

- Se inserta en la hoja más a la derecha, y hacemos swappings hasta encontrar el lugar correspondiente.

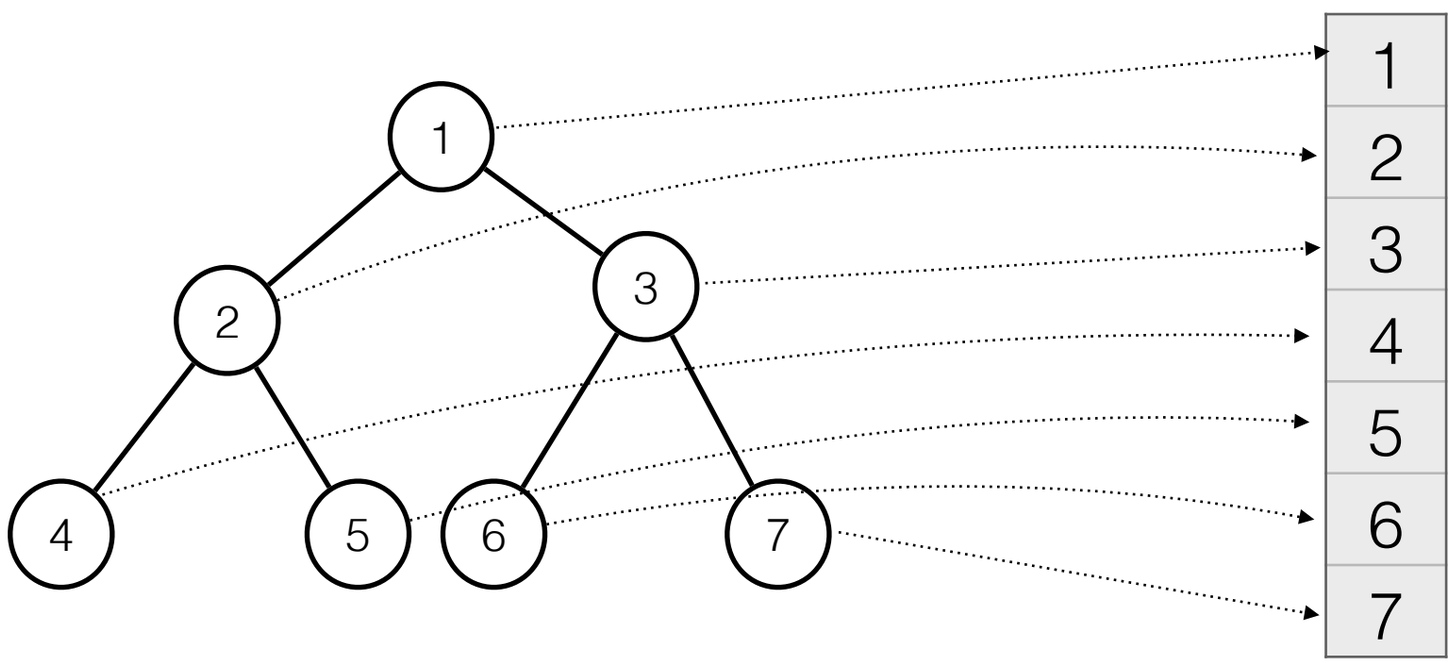
- La inserción es O(log n), se utiliza un referencia a la hoja más a la derecha, y luego tenemos log n intercambios.

- El borrado es similar y tenemos O(log n) intercambios.

Implementaciones de Heaps

Se puede lograr una implementación eficiente de heaps con arreglos:

- La posición 0 contiene la raíz,
- Para cada posición i, en 2*i+1 se encuentra su hijo izq., y en 2*(i+1) se encuentra su hijo derecho.



HeapSort

Podemos hacer un algoritmo de ordenación eficiente con heaps:

- Se recorre el arreglo a ordenar insertando cada elemento en un heap,
- Se van sacando uno a uno los elementos del heap, insertandolos en un nuevo arreglo,
- El arreglo que se obtiene está ordenado!

El tiempo de ejecución del algoritmo es O(n*log n), eficiente en la práctica.

Otras Operaciones sobre Heaps

Hay varias otras operaciones sobre heaps que se pueden hacer eficientemente:

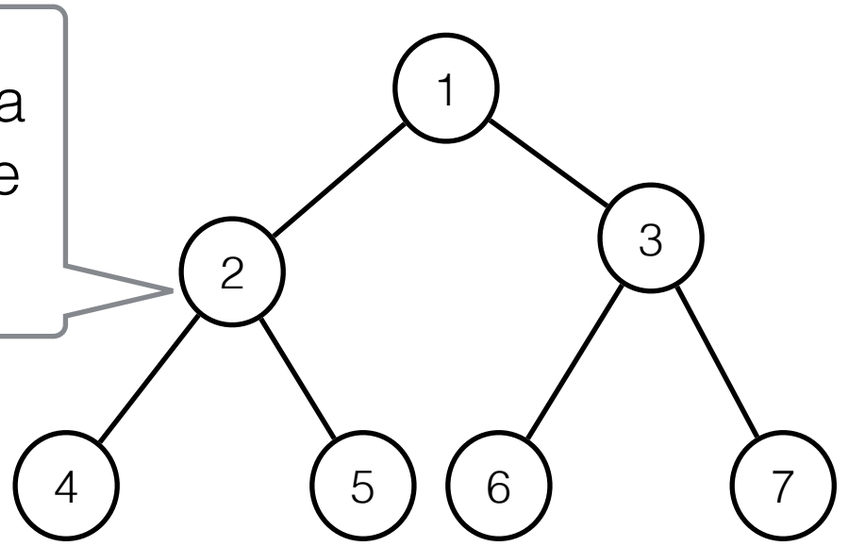
increaseKey(int pos, Number delta)

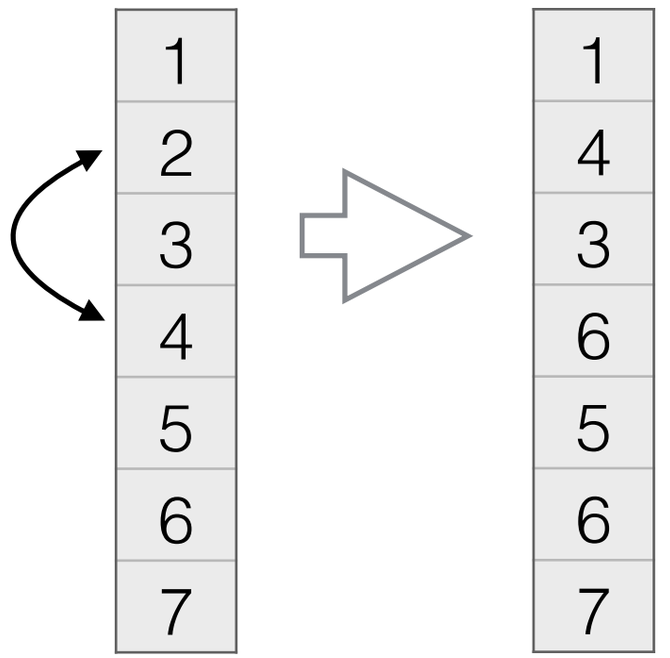
Se puede implementar facilmente con arreglos

Incrementa la clave del elemento en la posición pos, en delta

Se puede hacer en O(log n)

Si sumamos 6 a 2, tenemos que bajarlo





Otras Operaciones sobre Heaps

decreaseKey(int pos, Number delta)

si conocemos la posición de un elemento podemos decrementar su valor y acomodarlo

Si restamos 4 a 4, tenemos que subirlo

Se puede hacer en O(log n)